



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/643,727	08/18/2003	James R. Kohn	1376.731US1	4686

21186 7590 08/11/2006

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. BOX 2938
MINNEAPOLIS, MN 55402

EXAMINER

FENNEMA, ROBERT E

ART UNIT PAPER NUMBER

2183

DATE MAILED: 08/11/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.		Applicant(s)	
	10/643,727		KOHN, JAMES R.	
	Examiner		Art Unit	
	Robert E. Fennema		2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 15 June 2006.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-29 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-6, 8-10, 12-14, 16-23 and 25-29 is/are rejected.
- 7) ☒ Claim(s) 7, 11, 15 and 24 is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 15 June 2006 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413) |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | Paper No(s)/Mail Date. _____ |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| Paper No(s)/Mail Date <u>2/21/06; 6/15/06</u> . | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. Claims 1-27 have been considered. Claim 23 has been amended as per Applicants request.

Allowable Subject Matter

2. Claims 7, 11, 15 and 24 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.
3. The following is a statement of reasons for the indication of allowable subject matter: While the applicant has disclosed in his admitted prior art that a sequence of values is written to and read back from memory to determine if duplicate addresses exist in a vector, the applicant has not disclosed in his admitted prior art how this sequence of values was determined in the conventional algorithms, nor has the method of determining as found in the applicants claims been found in any prior art.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Art Unit: 2183

5. Claims 1-4, 14, 16-18, and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Beard et al. (USPN 5,640,524, herein Beard), in view of Applicants admitted prior art (herein Kohn), further in view of Cray Assembly Language Systems Reference Manual (herein Cray).

6. As per Claim 1, Beard teaches: A computerized method comprising, in each of a plurality of processors including a first processor and a second processor (Column 4, Lines 42-45 disclose that while the preferred embodiment is done on a single processor, multiple processors can be used in another embodiment):

(a) loading a first vector register with addressing values (Column 2 Line 65 – Column 3 Line 1);

(b) loading a second vector register with operand values (Column 3, Lines 10-12.

The second vector instruction would necessarily need a vector of operand values in order to operate on the retrieved data words);

(e) loading, using indirect addressing from the first vector register, elements from memory into a third vector register (Column 3 Lines 5-10, while Column 28, Lines 18-19 show indirect addressing capabilities);

(f) adding values from the third vector register and the second vector of operand values to generate a result vector (Column 3, Lines 10-13); and

(g) storing the result vector to memory using indirect addressing (Column 3, Lines 11-13, while Column 28, Lines 18-19 show indirect addressing capabilities), but fails to teach:

Art Unit: 2183

(c) determining which, if any, element addresses of the first vector register have a value that duplicates a value in another element address.

However, Kohn has taught that in the prior art (the "conventional" algorithm as described on pages 4 and 5 of the specification) that after loading from memory, you would store a known pattern, load it back, and compare against the original pattern to determine collisions. Kohn further teaches that by doing this, not only would you see if there was a collision, but also which elements were colliding (Page 5). Given this disclosure, it would have been obvious to one of ordinary skill in the art at the time the invention was made to store a sequence of values to memory, read it back out, and comparing the two to determine which elements did not compare correctly (and to do so, would require a circuit to do so), to allow correct execution of the instructions, which is critical for computers.

Beard further fails to teach:

(d) selectively adding certain elements of the second vector of operand values based on the element addresses the duplicated values.

However, Cray teaches an instruction set that runs vector instructions and supports a vector architecture, one of which would have been obvious to run on Beard's invention, as Beard's invention is an improvement of the original Cray vector processor, as disclosed in Column 1 Line 66- Column 2, Line 6) and thus would be compatible, as every system needs some instruction set architecture, and Beard does not teach a specific instruction set to use in his invention. One of the instructions Cray teaches is a vector compress (Section 2.5.4), which uses a vector as a bit mask to modify a register

Art Unit: 2183

to: invalidate certain elements, modify elements (if the destination is the same as source), and can also load a new register with elements (if the destination is different than the source), which is advantageous for improving the performance of decoupled execution, as also described in Section 2.5.4. Beard also teaches that an example arithmetic instruction that could be run on a vector is an add (Column 5, Lines 3-10). When a vector register is masked, and added to itself, then it has elements which are combined based on results of the comparing (when the comparison results are used as the mask). Therefore, one of ordinary skill in the art at the time the invention was made would have recognized the improvement of decoupled execution of using mask and compression instructions, and would have modified Beards invention to include and utilize them to gain those performance advantages.

7. As per Claim 2, Beard teaches: The method of claim 1, wherein the set of operations (a),(b),(c), and (d) is performed substantially in parallel in the plurality of processors, and the set of operations (e), (f), and (g) is performed serially, one processor at a time (Column 4, Lines 39-49 teach how the "resource monitoring and dependant initiation methods" could be implemented in minimally parallel processors. It appears that steps (a) through (d) are initiation steps towards setting up a load and then a result calculation, and thus would have been capable of being performed in parallel in Beards invention. As for steps (e) through (g), they necessarily must have been performed serially in order, as each step depends on the step preceding it in order to generate a correct output).

8. As per Claim 3, Beard teaches: The method of claim 1, further comprising executing an ordered Msync operation before the set of operations (e), (f), and (g); (It is necessary for steps (e) through (g) to execute in order, in order to get correct output, and it also is necessary for steps (a) through (d) to be completed prior, in order for the correct values to be available to step (e). One of ordinary skill in the art would recognize that to do this, an Msync operation could be run before step (e) to ensure the processor is in the correct state, with the memory contained all the necessary information. See the Msync description from "The Single UNIX Specification", which states that Msync should be used by programs that require a memory object to be in a known state) and

executing an end ordered Msync operation after the set of operations (e), (f), and (g) (After all the required steps are completed, one of ordinary skill in the art would have recognized the need to write that state to memory, so that any other processes (using the other processors in the multi-processor environment) waiting for that data can have it made available to them).

9. As per Claim 4, Beard teaches: The method of claim 3, wherein the set of operations (a), (b), (c), and (d) is performed substantially in parallel in the plurality of processors (Column 4, Lines 39-49 teach how the "resource monitoring and dependant initiation methods" could be implemented in minimally parallel processors. It appears that steps (a) through (d) are initiation steps towards setting up a load and then a result

Art Unit: 2183

calculation, and thus would have been capable of being performed in parallel in Beards invention).

10. As per Claim 14, Beard teaches: A system comprising;

a first vector register having addressing values (Column 2 Line 65 – Column 3 Line 1);

a second vector register having operand values (Column 3, Lines 10-12. The second vector instruction would necessarily need a vector of operand values in order to operate on the retrieved data words);

circuitry programmed to load, using indirect addressing from the first vector register, elements from memory into a third vector register (Column 3 Lines 5-10, while Column 28, Lines 18-19 show indirect addressing capabilities);

circuitry programmed to add values from the third vector register and the second vector of operand values to generate a result vector (Column 3, Lines 10-13); and

circuitry programmed to store the result vector to memory using indirect addressing (Column 3 Lines 11-13, while Column 28, Lines 18-19 show indirect addressing capabilities), but fails to teach:

circuitry programmed to determine which, if any, element addresses of the first vector register have a value that duplicates a value in another element address.

However, Kohn has taught that in the prior art (the “conventional” algorithm as described on pages 4 and 5 of the specification) that after loading from memory, you would store a known pattern, load it back, and compare against the original pattern to

Art Unit: 2183

determine collisions. Kohn further teaches that by doing this, not only would you see if there was a collision, but also which elements were colliding (Page 5). Given this disclosure, it would have been obvious to one of ordinary skill in the art at the time the invention was made to store a sequence of values to memory, read it back out, and comparing the two to determine which elements did not compare correctly (and to do so, would require a circuit to do so), to allow correct execution of the instructions, which is critical for computers.

Beard further fails to teach:

circuitry programmed to selectively add certain elements of the second vector of operand values based on the element addresses the duplicated values.

However, Cray teaches an instruction set that runs vector instructions and supports a vector architecture, one of which would have been obvious to run on Beard's invention, as Beard's invention is an improvement of the original Cray vector processor, as disclosed in Column 1 Line 66- Column 2, Line 6) and thus would be compatible, as every system needs some instruction set architecture, and Beard does not teach a specific instruction set to use in his invention. One of the instructions Cray teaches is a vector compress (Section 2.5.4), which uses a vector as a bit mask to modify a register to: invalidate certain elements, modify elements (if the destination is the same as source), and can also load a new register with elements (if the destination is different than the source), which is advantageous for improving the performance of decoupled execution, as also described in Section 2.5.4. Beard also teaches that an example arithmetic instruction that could be run on a vector is an add (Column 5, Lines 3-10).

When a vector register is masked, and added to itself, then it has elements which are combined based on results of the comparing (when the comparison results are used as the mask). Therefore, one of ordinary skill in the art at the time the invention was made would have recognized the improvement of decoupled execution of using mask and compression instructions, and would have modified Beards invention to include and utilize them to gain those performance advantages.

11. As per Claim 16, Beard teaches: The system of claim 14, further comprising:

circuitry programmed to perform the set of operations (a), (b), (c), and (d) substantially in parallel in the plurality of processors, and

circuitry programmed to perform the set of operations (e), (f), and (g) serially, one processor at a time (Column 4, Lines 39-49 teach how the "resource monitoring and dependant initiation methods" could be implemented in minimally parallel processors. It appears that steps (a) through (d) are initiation steps towards setting up a load and then a result calculation, and thus would have been capable of being performed in parallel in Beards invention. As for steps (e) through (g), they necessarily must have been performed serially in order, as each step depends on the step preceding it in order to generate a correct output).

12. As per Claim 17, Beard teaches: The system of claim 14, further comprising:

circuitry programmed to execute an ordered Msync operation before the set of operations (e), (f), and (g) (It is necessary for steps (e) through (g) to execute in order,

Art Unit: 2183

in order to get correct output, and it also is necessary for steps (a) through (d) to be completed prior, in order for the correct values to be available to step (e). One of ordinary skill in the art would recognize that to do this, an Msync operation could be run before step (e) to ensure the processor is in the correct state, with the memory contained all the necessary information. See the Msync description from "The Single UNIX Specification", which states that Msync should be used by programs that require a memory object to be in a known state); and

circuitry programmed to execute an end ordered Msync operation after the set of operations (e), (f), and (g) (After all the required steps are completed, one of ordinary skill in the art would have recognized the need to write that state to memory, so that any other processes (using the other processors in the multi-processor environment) waiting for that data can have it made available to them).

13. As per Claim 18, Beard teaches: The system of claim 17, further comprising:

circuitry programmed to perform the set of operations (a), (b), (c), and (d) substantially in parallel in the plurality of processors (Column 4, Lines 39-49 teach how the "resource monitoring and dependant initiation methods" could be implemented in minimally parallel processors. It appears that steps (a) through (d) are initiation steps towards setting up a load and then a result calculation, and thus would have been capable of being performed in parallel in Beards invention).

Art Unit: 2183

14. As per Claim 27, Beard teaches: A computer-readable medium having instructions stored thereon for causing a suitably programmed information-processing system to execute a method comprising:

loading a first vector register with addressing values (Column 2 Line 65 – Column 3 Line 1);

loading a second vector register with operand values (Column 3, Lines 10-12. The second vector instruction would necessarily need a vector of operand values in order to operate on the retrieved data words);

loading, using indirect addressing from the first vector register, elements from memory into a third vector register (Column 3 Lines 5-10, while Column 28, Lines 18-19 show indirect addressing capabilities);

adding values from the third vector register and the second vector of operand values to generate a result vector (Column 3, Lines 10-13); and

storing the result vector to memory using indirect addressing (Column 3 Lines 11-13, while Column 28, Lines 18-19 show indirect addressing capabilities), but fails to teach:

determining which, if any, element addresses of the first vector register have a value that duplicates a value in another element address.

However, Kohn has taught that in the prior art (the “conventional” algorithm as described on pages 4 and 5 of the specification) that after loading from memory, you would store a known pattern, load it back, and compare against the original pattern to determine collisions. Kohn further teaches that by doing this, not only would you see if

Art Unit: 2183

there was a collision, but also which elements were colliding (Page 5). Given this disclosure, it would have been obvious to one of ordinary skill in the art at the time the invention was made to store a sequence of values to memory, read it back out, and comparing the two to determine which elements did not compare correctly (and to do so, would require a circuit to do so), to allow correct execution of the instructions, which is critical for computers.

Beard further fails to teach:

selectively adding certain elements of the second vector of operand values based on the element addresses the duplicated values.

However, Cray teaches an instruction set that runs vector instructions and supports a vector architecture, one of which would have been obvious to run on Beard's invention, as Beard's invention is an improvement of the original Cray vector processor, as disclosed in Column 1 Line 66- Column 2, Line 6) and thus would be compatible, as every system needs some instruction set architecture, and Beard does not teach a specific instruction set to use in his invention. One of the instructions Cray teaches is a vector compress (Section 2.5.4), which uses a vector as a bit mask to modify a register to: invalidate certain elements, modify elements (if the destination is the same as source), and can also load a new register with elements (if the destination is different than the source), which is advantageous for improving the performance of decoupled execution, as also described in Section 2.5.4. Beard also teaches that an example arithmetic instruction that could be run on a vector is an add (Column 5, Lines 3-10). When a vector register is masked, and added to itself, then it has elements which are

Art Unit: 2183

combined based on results of the comparing (when the comparison results are used as the mask). Therefore, one of ordinary skill in the art at the time the invention was made would have recognized the improvement of decoupled execution of using mask and compression instructions, and would have modified Beards invention to include and utilize them to gain those performance advantages.

15. Claims 5-6, 8-10, 12-13, 19-23, and 25-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Beard, Kohn, and Cray, further in view of Patterson et al. (herein Patterson).

16. As per Claim 5, Beard teaches the method of claim 1, but fails to teach: further comprising

executing a first barrier synchronization operation before the set of operations (e), (f), and (g) in all of the plurality of processors;

executing a second barrier synchronization operation before the set of operations (e), (f) and (g) in the second processor;

executing the set of operations (e), (f), and (g) in the first processor and then executing a second barrier synchronization operation in the first processor to satisfy the second barrier synchronization in the second processor, and executing a third barrier synchronization in the first processor; and

executing the set of operations (e), (f), and (g) in the second processor and then executing a third barrier synchronization operation in the second processor to satisfy the third barrier synchronization in the first processor.

It appears necessary that steps (a) through (d) be completely prior to steps (e) through (g) executing, and that steps (e) through (g) be done in order, in order to generate the correct output. Patterson teaches that in multiprocessor systems running parallel operations on the same data, a barrier is a common synchronization technique to force concurrently running processes to wait for previous instructions (or steps) to complete. One of ordinary skill in the art would have recognized this method as a common way to synchronize multiple processors working on these steps, and how to use them. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use a barrier to ensure that the steps were done in the appropriate order.

17. As per Claim 6, Beard teaches: The method of claim 5, wherein the set of operations (a), (b), (c), and (d) is performed substantially in parallel in the plurality of processors (Column 4, Lines 39-49 teach how the "resource monitoring and dependant initiation methods" could be implemented in minimally parallel processors. It appears that steps (a) through (d) are initiation steps towards setting up a load and then a result calculation, and thus would have been capable of being performed in parallel in Beards invention).

18. As per Claim 8, Beard teaches: A computerized method comprising:

(a) within a first vector processor:

loading a first vector register in the first vector processor with addressing values (Column 2 Line 65 – Column 3 Line 1);

loading a second vector register in the first vector processor with operand values (Column 3, Lines 10-12. The second vector instruction would necessarily need a vector of operand values in order to operate on the retrieved data words);

selectively adding certain elements of the second vector of operand values in the first vector processor based on the element addresses the duplicated values (Column 3, Lines 10-13, where addition is a common operation that can be used);

(b) within a second vector processor:

loading a first vector register in the second vector processor with addressing values (Column 2 Line 65 – Column 3 Line 1);

loading a second vector register in the second vector processor with operand values (Column 3, Lines 10-12. The second vector instruction would necessarily need a vector of operand values in order to operate on the retrieved data words);

(d) within the first vector processor:

loading, using indirect addressing from the first vector register, elements from memory into a third vector register in the first vector processor (Column 3 Lines 5-10, while Column 28, Lines 18-19 show indirect addressing capabilities);

Art Unit: 2183

operating on values from the third vector register and the second vector of operand values in the first vector processor to generate a first result vector (Column 3, Lines 10-13); and

storing the first result vector to memory using indirect addressing (Column 3, Lines 11-13, while Column 28, Lines 18-19 show indirect addressing capabilities);

(f) within the second vector processor:

loading, using indirect addressing from the first vector register, elements from memory into a third vector register in the second vector processor (Column 3 Lines 5-10, while Column 28, Lines 18-19 show indirect addressing capabilities);

operating on values from the third vector register and the second vector of operand values in the second vector processor to generate a second result vector (Column 3, Lines 10-13); and

storing the second result vector to memory using indirect addressing (Column 3, Lines 11-13, while Column 28, Lines 18-19 show indirect addressing capabilities), but fails to teach:

determining which, if any, element addresses of the first vector register in the first vector processor have a value that duplicates a value in another element address;

determining which, if any, element addresses of the first vector register in the second vector processor have a value that duplicates a value in another element address.

However, Kohn has taught that in the prior art (the "conventional" algorithm as described on pages 4 and 5 of the specification) that after loading from memory, you

Art Unit: 2183

would store a known pattern, load it back, and compare against the original pattern to determine collisions. Kohn further teaches that by doing this, not only would you see if there was a collision, but also which elements were colliding (Page 5). Given this disclosure, it would have been obvious to one of ordinary skill in the art at the time the invention was made to store a sequence of values to memory, read it back out, and comparing the two to determine which elements did not compare correctly (and to do so, would require a circuit to do so), to allow correct execution of the instructions, which is critical for computers.

Beard further fails to teach:

selectively operating on certain elements of the second vector of operand values in the second vector processor based on the element addresses the duplicated values.

However, Cray teaches an instruction set that runs vector instructions and supports a vector architecture, one of which would have been obvious to run on Beard's invention, as Beard's invention is an improvement of the original Cray vector processor, as disclosed in Column 1 Line 66- Column 2, Line 6) and thus would be compatible, as every system needs some instruction set architecture, and Beard does not teach a specific instruction set to use in his invention. One of the instructions Cray teaches is a vector compress (Section 2.5.4), which uses a vector as a bit mask to modify a register to: invalidate certain elements, modify elements (if the destination is the same as source), and can also load a new register with elements (if the destination is different than the source), which is advantageous for improving the performance of decoupled execution, as also described in Section 2.5.4. Beard also teaches that an example

Art Unit: 2183

arithmetic instruction that could be run on a vector is an add (Column 5, Lines 3-10).

When a vector register is masked, and added to itself, then it has elements which are combined based on results of the comparing (when the comparison results are used as the mask). Therefore, one of ordinary skill in the art at the time the invention was made would have recognized the improvement of decoupled execution of using mask and compression instructions, and would have modified Beards invention to include and utilize them to gain those performance advantages.

Beard further fails to teach:

- (c) performing a synchronization operation that ensures that prior store operations effectively complete in at least the second vector processor before the following (d) operation;
- (e) performing a synchronization operation that ensures that the storing of the first result vector effectively completes before the following (f) operations.

Patterson teaches that in multiprocessor systems running parallel operations on the same data, a barrier is a common synchronization technique to force concurrently running processes to wait for previous instructions (or steps) to complete. One of ordinary skill in the art would have recognized this method as a common way to synchronize multiple processors working on these steps, and how to use them. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use a barrier to ensure that the steps were done in the appropriate order.

19. As per Claim 9, Beard teaches: The method of claim 8, wherein each of the operating on functions includes adding (Column 11, Line 32, where an adder adds).

20. As per Claim 10, Beard teaches: The method of claim 9, wherein the adding includes a floating-point addition operation that produces at least one element of the result vector as an ordered- operation floating point summation of an element of the loaded third vector register and a plurality of respective elements of the original second vector of operand values corresponding to elements of the first vector of addressing values having identical values (Column 4 Line 65 – Column 5 Line 19 shows floating point to be one of the available operations. Also see Column 3, Lines 10-13).

21. As per Claim 12, Cray teaches: The method of claim 8, further wherein the loading of the third vector register of each processor includes loading elements from locations specified by addressing values corresponding to indications of positive compares from the comparing operation (Column 3, Lines 5-12. Beard teaches that a vector is loaded based on the addressing values of the first addressing vector register, but does not teach that the elements are loaded only from elements which indicated a compare. However, looking at the cmpress() function from Cray as described in Section 2.5.4, when the first addressing register is masked, only elements that have been indicated as a compare (if mj[I]) are stored in the destination register (the masked addressing register), and the others will be set as undefined, thus will be unable to load values).

22. As per Claim 13, Beard teaches: The method of claim 8, further wherein indirect addresses of the elements from memory are calculated by adding each respective addressing value to a base address (Column 3 Lines 2-5).

23. As per Claim 19, Beard teaches the system of claim 14, but fails to teach: further comprising:

circuitry programmed to execute a first barrier synchronization operation before the set of operations (e), (f), and (g) in all of the plurality of processors;

circuitry programmed to execute a second barrier synchronization operation before the set of operations (e), (f), and (g) in the second processor;

circuitry programmed to execute the set of operations (e), (f), and (g) in the first processor and then executing a second barrier synchronization operation in the first processor to satisfy the second barrier synchronization in the second processor, and executing a third barrier synchronization in the first processor; and

circuitry programmed to execute the set of operations (e), (f), and (g) in the second processor and then executing a third barrier synchronization operation in the second processor to satisfy the third barrier synchronization in the first processor.

It appears necessary that steps (a) through (d) be completely prior to steps (e) through (g) executing, and that steps (e) through (g) be done in order, in order to generate the correct output. Patterson teaches that in multiprocessor systems running parallel operations on the same data, a barrier is a common synchronization technique

Art Unit: 2183

to force concurrently running processes to wait for previous instructions (or steps) to complete. One of ordinary skill in the art would have recognized this method as a common way to synchronize multiple processors working on these steps, and how to use them. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use a barrier to ensure that the steps were done in the appropriate order.

24. As per Claim 20, Beard teaches: The system of claim 19, further comprising circuitry programmed to perform the set of operations (a), (b), (c), and (d) substantially in parallel in the plurality of processors (Column 4, Lines 39-49 teach how the “resource monitoring and dependant initiation methods” could be implemented in minimally parallel processors. It appears that steps (a) through (d) are initiation steps towards setting up a load and then a result calculation, and thus would have been capable of being performed in parallel in Beards invention).

25. As per Claim 21, Beard teaches: A system comprising:

(a) a first vector processor that includes:

means for loading a first vector register in the first vector processor with addressing values (Column 2 Line 65 – Column 3 Line 1);

means for loading a second vector register in the first vector processor with operand values (Column 3, Lines 10-12. The second vector instruction would

Art Unit: 2183

necessarily need a vector of operand values in order to operate on the retrieved data words);

means for selectively adding certain elements of the second vector of operand values in the first vector processor based on the element addresses the duplicated values (Column 3, Lines 10-13, where addition is a common operation that can be used);

(b) a second vector processor that includes:

means for loading a first vector register in the second vector processor with addressing values (Column 2 Line 65 – Column 3 Line 1);

means for loading a second vector register in the second vector processor with operand values (Column 3, Lines 10-12. The second vector instruction would necessarily need a vector of operand values in order to operate on the retrieved data words);

(d) within the first vector processor;

means for loading, using indirect addressing from the first vector register, elements from memory into a third vector register in the first vector processor (Column 3 Lines 5-10, while Column 28, Lines 18-19 show indirect addressing capabilities);

means for operating on values from the third vector register and the second vector of operand values in the first vector processor to generate a first result vector (Column 3, Lines 10-13); and

means for storing the first result vector to memory using indirect addressing;

Art Unit: 2183

(f) within the second vector processor (Column 3, Lines 11-13, while Column 28, Lines 18-19 show indirect addressing capabilities):

means for loading, using indirect addressing from the first vector register, elements from memory into a third vector register in the second vector processor (Column 3 Lines 5-10, while Column 28, Lines 18-19 show indirect addressing capabilities);

means for operating on values from the third vector register and the second vector of operand values in the second vector processor to generate a second result vector (Column 3, Lines 10-13); and

means for storing the second result vector to memory using indirect addressing (Column 3, Lines 11-13, while Column 28, Lines 18-19 show indirect addressing capabilities), but fails to teach:

means for determining which, if any, element addresses of the first vector register in the first vector processor have a value that duplicates a value in another element address;

means for determining which, if any, element addresses of the first vector register in the second vector processor have a value that duplicates a value in another element address;

However, Kohn has taught that in the prior art (the "conventional" algorithm as described on pages 4 and 5 of the specification) that after loading from memory, you would store a known pattern, load it back, and compare against the original pattern to determine collisions. Kohn further teaches that by doing this, not only would you see if

Art Unit: 2183

there was a collision, but also which elements were colliding (Page 5). Given this disclosure, it would have been obvious to one of ordinary skill in the art at the time the invention was made to store a sequence of values to memory, read it back out, and comparing the two to determine which elements did not compare correctly (and to do so, would require a circuit to do so), to allow correct execution of the instructions, which is critical for computers.

Beard further fails to teach:

means for selectively operating on certain elements of the second vector of operand values in the second vector processor based on the element addresses the duplicated values.

However, Cray teaches an instruction set that runs vector instructions and supports a vector architecture, one of which would have been obvious to run on Beard's invention, as Beard's invention is an improvement of the original Cray vector processor, as disclosed in Column 1 Line 66- Column 2, Line 6) and thus would be compatible, as every system needs some instruction set architecture, and Beard does not teach a specific instruction set to use in his invention. One of the instructions Cray teaches is a vector compress (Section 2.5.4), which uses a vector as a bit mask to modify a register to: invalidate certain elements, modify elements (if the destination is the same as source), and can also load a new register with elements (if the destination is different than the source), which is advantageous for improving the performance of decoupled execution, as also described in Section 2.5.4. Beard also teaches that an example arithmetic instruction that could be run on a vector is an add (Column 5, Lines 3-10).

Art Unit: 2183

When a vector register is masked, and added to itself, then it has elements which are combined based on results of the comparing (when the comparison results are used as the mask). Therefore, one of ordinary skill in the art at the time the invention was made would have recognized the improvement of decoupled execution of using mask and compression instructions, and would have modified Beards invention to include and utilize them to gain those performance advantages.

Beard further fails to teach:

(c) means for performing a synchronization operation that ensures that prior store operations effectively complete in at least the second vector processors before the operations of the following (d) means;

(e) performing a synchronization operation that ensures that the storing of the first result vector effectively completes before the operations of the following (f) means.

Patterson teaches that in multiprocessor systems running parallel operations on the same data, a barrier is a common synchronization technique to force concurrently running processes to wait for previous instructions (or steps) to complete. One of ordinary skill in the art would have recognized this method as a common way to synchronize multiple processors working on these steps, and how to use them. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use a barrier to ensure that the steps were done in the appropriate order.

Art Unit: 2183

26. As per Claim 22, Beard teaches: The system of claim 21, wherein each of the means for operating on functions includes an adder (Column 11, Line 32).

27. As per Claim 23, Beard teaches: The system of claim 22, further wherein the adder includes a floating-point adder that produces at least one element of the result vector as an ordered-operation floating point summation of an element of the loaded third vector register and a plurality of respective elements of the original second vector of operand values corresponding to elements of the first vector of addressing values having identical values (Column 4 Line 65 – Column 5 Line 19 shows floating point to be one of the available operations. Also see Column 3, Lines 10-13).

28. As per Claim 25, Cray teaches: The system of claim 21, further wherein the means for loading of the third vector register of each processor includes means for loading elements from locations specified by addressing values corresponding to indications of positive compares from the comparing operation (Column 3, Lines 5-12. Beard teaches that a vector is loaded based on the addressing values of the first addressing vector register, but does not teach that the elements are loaded only from elements which indicated a compare. However, looking at the `compress()` function from Cray as described in Section 2.5.4, when the first addressing register is masked, only elements that have been indicated as a compare (if `mj[I]`) are stored in the destination register (the masked addressing register), and the others will be set as undefined, thus will be unable to load values).

29. As per Claim 26, Beard teaches: The system of claim 21, further wherein indirect addresses of the elements from memory are calculated by adding each respective addressing value to a base address (Column 3 Lines 2-5).

Response to Arguments

30. Examiner makes note of the amended specification, Claim 23, and the new drawings, and withdraws the objection to each.

31. Applicant's arguments, see Page 1, filed 6/15/2006, with respect to 112 Rejection of the Claims have been fully considered and are persuasive. The rejection of Claims 7, 11, 15, and 24 have been withdrawn.

32. Applicant's arguments with respect to claims 1-27 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Robert E. Fennema whose telephone number is (571) 272-2748. The examiner can normally be reached on Monday-Friday, 8:00-5:30.

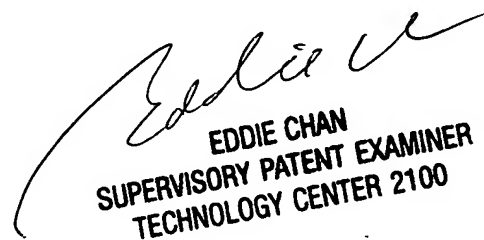
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Art Unit: 2183

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Robert E Fennema
Examiner
Art Unit 2183

RF



EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100